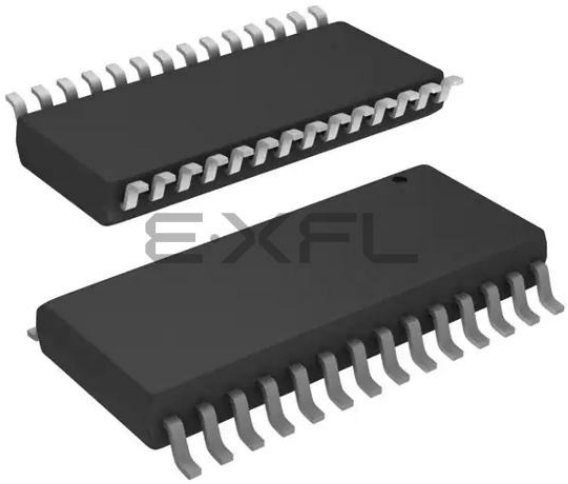


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	21
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25j10t-i-so

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets	Wake-up via WDT or Interrupt
TRISE	PIC18F2XJ10	PIC18F4XJ10	0000 -111	1111 -111	uuuu -uuu
TRISD	PIC18F2XJ10	PIC18F4XJ10	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F2XJ10	PIC18F4XJ10	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F2XJ10	PIC18F4XJ10	1111 1111	1111 1111	uuuu uuuu
TRISA	PIC18F2XJ10	PIC18F4XJ10	--1- 1111	--1- 1111	--u- uuuu
SSP2BUF	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE	PIC18F2XJ10	PIC18F4XJ10	---- -xxx	---- -uuu	---- -uuu
LATD	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	PIC18F2XJ10	PIC18F4XJ10	--xx xxxx	--uu uuuu	--uu uuuu
SSP2ADD	PIC18F2XJ10	PIC18F4XJ10	0000 0000	0000 0000	uuuu uuuu
SSP2STAT	PIC18F2XJ10	PIC18F4XJ10	0000 0000	0000 0000	uuuu uuuu
SSP2CON1	PIC18F2XJ10	PIC18F4XJ10	0000 0000	0000 0000	uuuu uuuu
SSP2CON2	PIC18F2XJ10	PIC18F4XJ10	0000 0000	0000 0000	uuuu uuuu
PORTE	PIC18F2XJ10	PIC18F4XJ10	---- -xxx	---- -uuu	---- -uuu
PORTD	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F2XJ10	PIC18F4XJ10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	PIC18F2XJ10	PIC18F4XJ10	--0- 0000	--0- 0000	--u- uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 5-1 for Reset value for specific condition.

6.1.3 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 6.1.6.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

6.1.4 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the top-of-stack Special Function Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

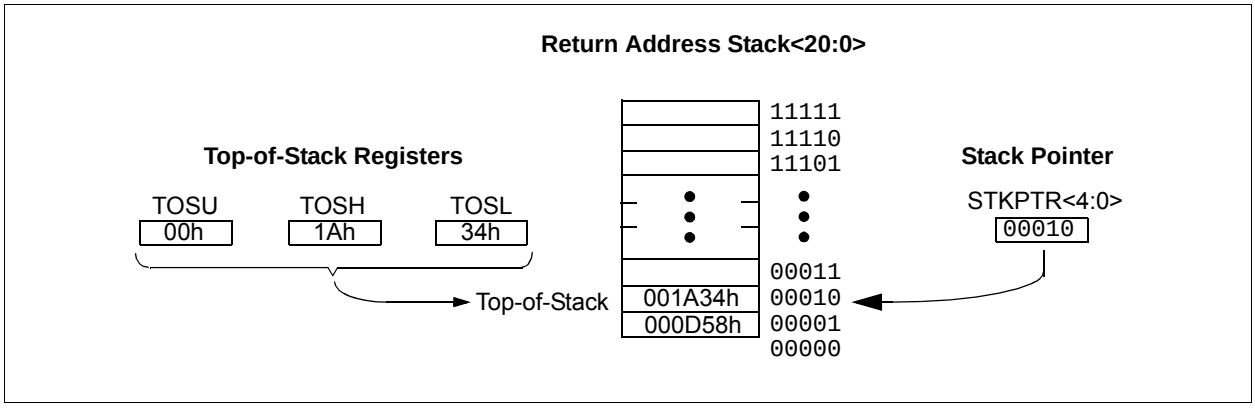
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

6.1.4.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 6-3). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses indirect addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

6.5 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remains unchanged.

6.5.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = 0) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

6.5.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘1’), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled in shown in Figure 6-9.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 22.2.1 “Extended Instruction Syntax”**.

12.1 Timer1 Operation

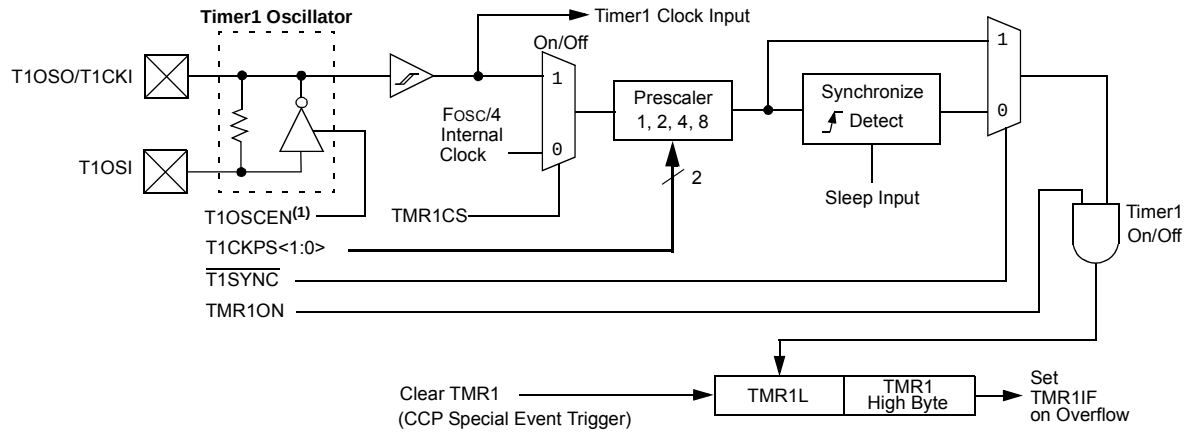
Timer1 can operate in one of these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR1CS is cleared (= 0), Timer1 increments on every internal instruction cycle (FOSC/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

FIGURE 12-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

15.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L register contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The PWM duty cycle is calculated by the following equation:

EQUATION 15-2:

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into CCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation. When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the CCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation:

EQUATION 15-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

15.4.3 PWM OUTPUT CONFIGURATIONS

The P1M<1:0> bits in the CCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

The Single Output mode is the standard PWM mode discussed in **Section 15.4 "Enhanced PWM Mode"**. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

The general relationship of the outputs in all configurations is summarized in Figure 15-2.

TABLE 15-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

TABLE 16-4: REGISTERS ASSOCIATED WITH I²C™ OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	49
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	49
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	49
PIR2	OSCFIF	CMIF	—	—	BCL1IF	—	—	CCP2IF	49
PIE2	OSCFIE	CMIE	—	—	BCL1IE	—	—	CCP2IE	49
IPR2	OSCFIP	CMIP	—	—	BCL1IP	—	—	CCP2IP	49
PIR3	SSP2IF	BCL2IF	—	—	—	—	—	—	49
PIE3	SSP2IE	BCL2IE	—	—	—	—	—	—	49
IPR3	SSP2IP	BCL2IP	—	—	—	—	—	—	49
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	50
TRISD(1)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	50
SSP1BUF	MSSP1 Receive Buffer/Transmit Register								48
SSP1ADD	MSSP1 Address Register (I²C™ Slave mode). MSSP1 Baud Rate Reload Register (I²C Master mode).								48
SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	48
SSP1CON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	48
	GCEN	ACKSTAT	ADMSK5(2)	ADMSK4(2)	ADMSK3(2)	ADMSK2(2)	ADMSK1(2)	SEN	48
SSP1STAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	48
SSP2BUF	MSSP2 Receive Buffer/Transmit Register								50
SSP2ADD	MSSP2 Address Register (I²C Slave mode). MSSP2 Baud Rate Reload Register (I²C Master mode).								50
SSP2CON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	50
SSP2CON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	50
	GCEN	ACKSTAT	ADMSK5(2)	ADMSK4(2)	ADMSK3(2)	ADMSK2(2)	ADMSK1(2)	SEN	48
SSP2STAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	50

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the MSSP module in I²C™ mode.

Note 1: These registers and/or bits are not implemented on 28-pin devices and should be read as '0'.

2: Alternate names and definitions for these bits when the MSSP module is operating in I²C Slave mode. See **Section 16.4.3.2 "Address Masking"** for details.

EXAMPLE 17-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate = FOSC/(64 ([SPBRGH:SPBRG] + 1))

Solving for SPBRGH:SPBRG:

X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

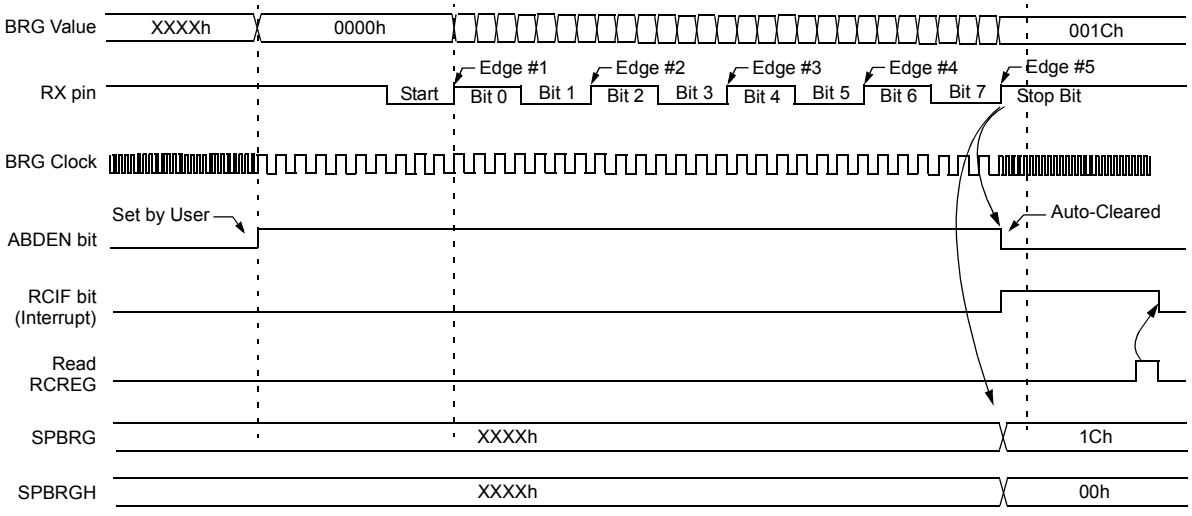
Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 17-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	49
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	49
SPBRGH	EUSART Baud Rate Generator Register High Byte								49
SPBRG	EUSART Baud Rate Generator Register Low Byte								49

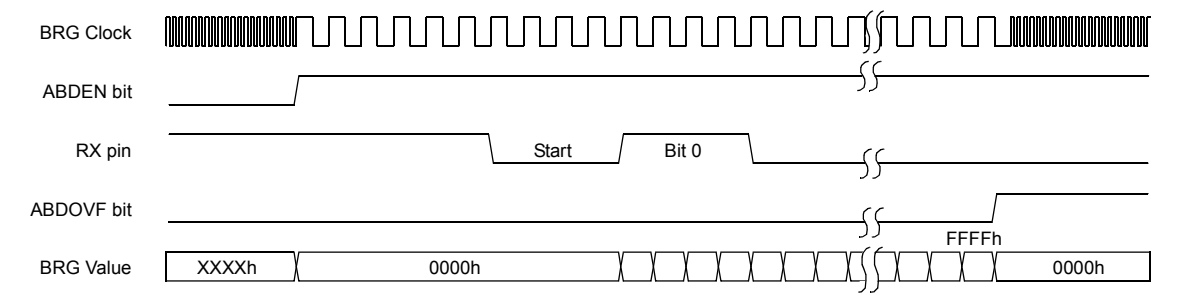
Legend: — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

FIGURE 17-1: AUTOMATIC BAUD RATE CALCULATION



Note: The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

FIGURE 17-2: BRG OVERFLOW SEQUENCE



REGISTER 18-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG<3:0>:** A/D Port Configuration Control bits:

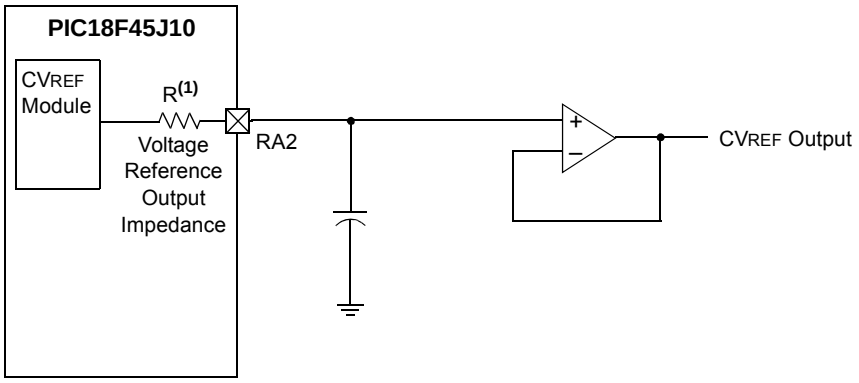
PCFG<3:0>	AN12	AN11	AN10	AN9	AN8	AN7(1)	AN6(1)	AN5(1)	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Note 1: AN5 through AN7 are available only on 40/44-pin devices.

NOTES:

FIGURE 20-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



Note 1: R is dependent upon the voltage reference configuration bits, CVRCON<3:0> and CVRCON<5>.

TABLE 20-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	49
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	49
TRISA	—	—	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0	50

Legend: Shaded cells are not used with the comparator voltage reference.

REGISTER 21-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

R/WO-1	R/WO-1	U-0	U-0	U-0	R/WO-1	R/WO-1	R/WO-1
IESO	FCMEN	—	—	—	FOSC2	FOSC1	FOSC0
bit 7							bit 0

Legend:		
R = Readable bit	WO = Write Once bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed	'1' = Bit is set	'0' = Bit is cleared

bit 7 **IESO:** Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit
1 = Two-Speed Start-up enabled
0 = Two-Speed Start-up disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5-3 **Unimplemented:** Read as '0'

bit 2 **FOSC2:** Default/Reset System Clock Select bit
1 = Clock selected by FOSC<1:0> as system clock is enabled when OSCCON<1:0> = 00
0 = INTRC enabled as system clock when OSCCON<1:0> = 00

bit 1-0 **FOSC<1:0>:** Oscillator Selection bits
11 = EC oscillator, PLL enabled and under software control, CLKO function on OSC2
10 = EC oscillator, CLKO function on OSC2
01 = HS oscillator, PLL enabled and under software control
00 = HS oscillator

REGISTER 21-5: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	WO = Write Once bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed	'1' = Bit is set	'0' = Bit is cleared

bit 7-0 **Unimplemented:** Read as '0'

REGISTER 21-6: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/WO-1
—(1)	—(1)	—(1)	—(1)	—	—	—	CCP2MX
bit 7							bit 0

Legend:		
R = Readable bit	WO = Write Once bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed	'1' = Bit is set	'0' = Bit is cleared

bit 7-1 **Unimplemented:** Read as '1'(1)

bit 0 **CCP2MX:** CCP2 MUX bit
1 = CCP2 is multiplexed with RC1
0 = CCP2 is multiplexed with RB3

Note 1: The value of these bits in program memory should always be '1'. This ensures that the location is executed as a NOP if it is accidentally executed.

BRA **Unconditional Branch**

Syntax: BRA n

Operands: $-1024 \le n \le 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1101	0nnn	nnnn	nnnn

Description: Add the 2's complement number, '2n', to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: HERE BRA Jump

Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

BSF **Bit Set f**

Syntax: BSF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $1 \rightarrow f<b>$

Status Affected: None

Encoding:

1000	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is set.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: BSF FLAG_REG, 7, 1

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

BTG **Bit Toggle f**

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: BOV n

Operands: $-128 \le n \le 127$

Operation: if Overflow bit is '1',
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number, '2n', is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example:
```
HERE    BOV   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE + 2)

SUBWFB **Subtract W from f with Borrow**

Syntax: SUBWFB f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (W) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0101	10da	ffff	ffff

Description: Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:
```
SUBWFB  REG, 1, 0
Before Instruction
    REG  =  19h    (0001 1001)
    W    =  0Dh    (0000 1101)
    C    =  1
After Instruction
    REG  =  0Ch    (0000 1011)
    W    =  0Dh    (0000 1101)
    C    =  1
    Z    =  0
    N    =  0      ; result is positive
```

Example 2:
```
SUBWFB  REG, 0, 0
Before Instruction
    REG  =  1Bh    (0001 1011)
    W    =  1Ah    (0001 1010)
    C    =  0
After Instruction
    REG  =  1Bh    (0001 1011)
    W    =  00h
    C    =  1
    Z    =  1      ; result is zero
    N    =  0
```

Example 3:
```
SUBWFB  REG, 1, 0
Before Instruction
    REG  =  03h    (0000 0011)
    W    =  0Eh    (0000 1101)
    C    =  1
After Instruction
    REG  =  F5h    (1111 0100)
                   ; [2's comp]
    W    =  0Eh    (0000 1101)
    C    =  0
    Z    =  0
    N    =  1      ; result is negative
```

SWAPF **Swap f**

Syntax: SWAPF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

Status Affected: None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:
```
SWAPF   REG, 1, 0
Before Instruction
    REG  =  53h
After Instruction
    REG  =  35h
```

ADDWF — ADD W to Indexed (Indexed Literal Offset mode)

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01d0	kkkk	kkkk

Description: The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.
If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write to destination

Example: ADDWF [OFST], 0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

BSF — Bit Set Indexed (Indexed Literal Offset mode)

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

1000	bbb0	kkkk	kkkk

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

SETF — Set Indexed (Indexed Literal Offset mode)

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

0110	1000	kkkk	kkkk

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write register

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

24.2 DC Characteristics: Power-Down and Supply Current PIC18F24J10/25J10/44J10/45J10 (Industrial) PIC18LF24J10/25J10/44J10/45J10 (Industrial)

PIC18F45J10 Family (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**	
	Power-Down Current (IPD)(1)					
	All devices	19	104	µA	-40°C	VDD = 2.5V (**Sleep** mode)
		25	104	µA	+25°C	
		40	184	µA	+85°C	
	All devices	20	203	µA	-40°C	VDD = 3.3V (**Sleep** mode)
		25	203	µA	+25°C	
		45	289	µA	+85°C	

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

24.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 24-4: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)

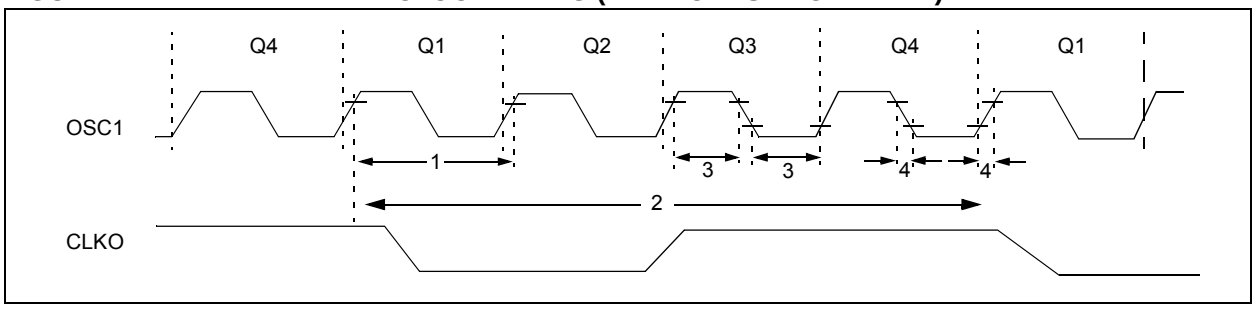


TABLE 24-6: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	40	MHz	EC Oscillator mode
		Oscillator Frequency(1)	4	25	MHz	HS Oscillator mode
1	TOSC	External CLKI Period(1)	25	—	ns	EC Oscillator mode
		Oscillator Period(1)	25	250	ns	HS Oscillator mode
2	TCY	Instruction Cycle Time(1)	100	—	ns	TCY = 4/FOSC, Industrial
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	10	—	ns	EC Oscillator mode
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	7.5	ns	EC Oscillator mode

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.